Data-Dependent Operator Selection Layers

Overview

We consider a variation on standard neural network layers where nonlinearities are used **only to make discrete decisions**, not to scale activations. These decisions are then used to select among linear operators.

At each layer, binary decisions $b_i(x) \in \{0,1\}$ are computed (e.g. from pre-activation sign tests). Instead of applying a ReLU, each bit selects between two matrices:

$$M_i(x) = \begin{cases} A_i & \text{if } b_i(x) = 1 \\ B_i & \text{if } b_i(x) = 0 \end{cases}$$

The layer output is then formed by combining these selected operators.

Layer Constructions

Two primary constructions:

1) Sum (ensemble-style)

$$y = \left(\sum_{i=1}^{n} M_i(x) \right) x$$

- Produces a **data-dependent linear operator** via summation
- Equivalent to selecting and combining (n) matrices per input

---

2) Product (composition-style)

$$y = \left(\prod_{i=1}^{n} M_i(x) \right) x$$

- Produces a **data-dependent composition of operators**
- Implicitly introduces depth within a single layer

---

3) Hybrids

Combinations of sums and products allow intermediate behavior:

- residual-style accumulation
- mixtures of shallow and deep operator paths

Comparison to Standard Layers

Standard ReLU layer:

y = W D(x) x

- D(x): diagonal gating (coordinate-wise)
- Each region of input space → one fixed linear map

Proposed layer:

$$y = \mathcal{T}(x)\, x$$

- $\mathcal{T}(x)$: selected from a combinatorial family of operators
- Each region → potentially very different transformation

Key difference:

- ReLU networks select **subspaces (via masks)**
- This approach selects **operators (via routing decisions)**

Functional Interpretation

The model can be viewed as implementing:

- a **discrete distribution over linear operators**
- indexed by binary decision patterns

This is related to:

- mixture-of-experts (hard routing)
- hypernetworks (input-dependent weights)
- decision-tree-like partitioning

but operates directly at the level of **full linear transformations**.

Expected Behavior

Sum variant

- Smooth interpolation between operators
- More stable optimization
- Tends toward averaging effects
- May exhibit spectral bias (favoring low-frequency components)

Product variant

- Highly expressive (combinatorial compositions)
- Implicit depth → strong nonlinear effects
- Sensitive to initialization and scaling
- Risk of:
    - exploding/vanishing norms
    - rank collapse

Practical Considerations

1) Training

- Binary decisions are non-differentiable
- Common strategies:
    - straight-through estimators
    - soft relaxations (sigmoid / Gumbel-softmax)

---

2) Parameterization

- Requires 2n matrices per layer
- Benefits from structure:
    - low-rank
    - orthogonal
    - circulant / fast transforms

---

3) Stability

Particularly important for the product variant:

- constrain operator norms
- use near-identity parameterizations:

  $A_i = I + U_i$, $B_i = I + V_i$

---

4) Regularization

- Encourage diversity in operator set
- Avoid collapse to identical matrices
- Control spectral properties (eigenvalue spread)

Conceptual Insight

This approach shifts the role of nonlinearity:

> Instead of modifying activations, nonlinearities select **which linear operator applies**.

This leads to:

- input-dependent transformations
- combinatorial operator reuse
- potentially more structured and interpretable behavior

Note on Transform Domains

While not required, it can be useful to express operators in a structured basis (e.g. diagonal in some orthogonal transform domain, such as the Fast Walsh–Hadamard Transform). In that setting, the model can be interpreted as selecting **input-dependent spectral filters**, which can improve efficiency and interpretability.

Summary

- Replace activation scaling with operator selection
- Use binary decisions to route between matrices
- Combine selected operators via sum or product
- Gain expressivity through combinatorial operator families
- Trade off stability vs power depending on construction

This provides an alternative design axis for neural networks:
operator routing instead of activation gating.